Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

SIMULATION OPTIMIZATION FOR A LARGE-SCALE BIKE-SHARING SYSTEM

Nanjing Jian
Daniel Freund
Holly M. Wiberg
Shane G. Henderson

Operations Research & Information Engineering
Cornell University
Ithaca, NY, 14853, U.S.A.

ABSTRACT

The Citi Bike system in New York City has approximately 466 stations, 6074 bikes, and 15777 docks. We wish to optimize both bike and dock allocations for each station at the beginning of the day, so that the expected number of customers who cannot find a bike, or cannot find a dock to return a bike, is minimized. With a system of this scale, traditional simulation optimization methods such as stochastic gradient-search and random search are inefficient. We propose a variety of more efficient gradient-like heuristic methods that can improve any given allocation based on a discrete-event simulation model of the system. The methods are tested on data from December 2015 with different starting solutions obtained from other models. We further explore the relationship between the system behaviors during the morning and afternoon rush hours by comparing optimal solutions when the problem is restricted to these two periods.

1 INTRODUCTION

Citi Bike in New York City (NYC) is a large-scale bike-sharing system where a customer can check out a bike from any station and return the bike to any other station. Due to time-varying demand, this freedom of movement of bikes creates unbalanced flows in the system. For example, many customers ride bikes from residential areas to the financial district during the morning rush (6-10am) and reverse their origin and destination in the afternoon. Citi Bike relocates bikes in the system, both overnight (12-6am) and throughout the day. Overnight rebalancing operations can move many more bikes than during the day due to traffic congestion. We are interested in how many docks should be assigned to each station and, based on that dock assignment, how many bikes should be allocated to each station at the beginning of the day.

This optimization problem is an extension of the rebalancing problem in the bike-sharing literature, in that we attempt to rebalance docks in addition to bikes. Insight into the rebalancing problem has been obtained by Fricker, Gast, and Mohamed (2012) and Fricker and Gast (2014) using mean-field analysis, showing, e.g., that in a system with perfectly balanced inflows and outflows at each station, that the optimal number of bikes is equal to half the combined station capacities plus the number on trips. Work that is more operational includes Henderson, O'Mahony, and Shmoys (2015), O'Mahony (2015) and O'Mahony (2015) where the CTMC model we refer to later is developed. Raviv and Kolka (2013) and Raviv, Tzur, and Forma (2013) develop a solution to bike (only) repositioning. Based on this work, Forma, Raviv, and Tzur (2015) develop a practical recipe for bike repositioning. Schuijbroek, Hampshire, and van Hoeve (2013) develop related integer programming models for repositioning bikes (only). Shu et al. (2013) use a time-space formulation that is more detailed than our models. Chemla, Meunier, and Calvo (2013), and Rainer-Harbach et al. (2013) are examples of other repositioning algorithms.

We use discrete-event simulation to model the bike-sharing system, and thus we tackle the rebalancing problem over bikes and docks as a simulation-optimization problem. Ideally, we would apply standard simulation-optimization methods to solve the problem, but as discussed in Jian and Henderson (2015) this seems computationally infeasible. In particular, the optimization problem (1) over 466 stations is an integer-ordered optimization problem with 932 decision variables. The scale of this problem renders methods based on random search principles unlikely to be effective. Indeed, since the numbers of bikes and docks are fixed, any local moves likely involve moving bikes and/or docks between randomly selected pairs of stations, and there are over 100,000 such pairs of stations. Moreover, if one attempts to perform a variant of gradient search where the gradients are estimated by finite (integer) differences, then each step of such an algorithm would require simulating 932 "neighboring" configurations (if forward differences are used).

These computational demands seem, to us, to be impractical. We instead develop heuristic search procedures that use statistics from a single simulation run in order to update the allocation of bikes and docks between stations. Our methods build on the heuristic introduced in Jian and Henderson (2015) for allocating bikes to satisfy only the morning rush demand, to allocate both bikes and docks for a full day's operations. We exploit the simulation and use station-specific information from it to group the stations according to their estimated contributions to the objective. This "contributions" can reasonably be viewed as *approximate* gradients, but we make no claim that our approximate gradients are accurate or unbiased, nor do we claim that we find a locally optimal solution. We instead see the value of these algorithms in the improvements they can make in performance relative to that of starting solutions.

In obtaining the station-specific information and generating a search direction, we are using the simulation as a "white box" that provides significantly more information than just the final performance estimate that is the starting point for more traditional "black box" approaches. By grouping stations using that information we have also reduced the dimensionality of the problem. These can be invaluable in search algorithms for large-scale simulation optimization problems.

We offer two primary contributions. First, we demonstrate a practical simulation-optimization approach to improving bike and dock allocations in bike systems that can tackle the large-scale instances seen in practice. Second, we (strive to) increase the attention of the simulation-optimization research community on large-scale practical problems that defy solution by existing simulation-optimization algorithms. In doing so, we hope to inspire algorithmic developments in simulation optimization on sub-classes of simulation-optimization problems.

2 PRELIMINARIES

This section introduces the problem and its input data, the discrete-event simulation model we use, and some alternatives for obtaining starting solutions for the simulation-optimization search. Parts of Sections 2.3, 2.4.2, and a time-homogeneous version of 2.4.3 are recaps of a discussion in Jian and Henderson (2015).

2.1 Problem Statement

Our goal is to minimize the expected number of "unhappy" customers who want to check out a bike when a station is empty or return a bike when a station is full, by giving an initial bike allocation x_i and dock (capacity) allocation r_i at each station i. We assume that the total number of bikes b and the total number of docks c is fixed in the system. The level of bikes x_i assigned is constrained by the capacity r_i, and the capacity r_i is constrained between 16 and 60 by physical space limits at the station. (This latter restriction is a simplification, since the actual space limits vary from station to station.) In practice in NYC, docks mostly come in sets of 3, but we ignore that complication in the work that follows. We use ξ to denote the random objects in the system, including the random arrivals and departures of customers at each station as well as the trip durations, so the notation accommodates for the use of Sample Average Approximation

(Kim, Pasupathy, and Henderson 2014). Thus the problem can be formulated as

$$\begin{aligned} \underset{x,r}{\text{minimize}} \quad & Ef(x,r;\xi) \\ \text{subject to} \quad & \sum_i x_i = b \\ & \sum_i r_i = c \\ & 0 \leq x_i \leq r_i, \ \forall i \\ & 16 \leq r_i \leq 60, \ \forall i \\ & x_i, r_i \text{ integer}, \ \forall i. \end{aligned} \tag{1}$$

The function f yields the exact number of unhappy customers in one day, and we estimate its expectation using simulation.

2.2 Input Data

We use real trip data from the 14 weekdays in the period December 1 to 20, 2015. We selected this time period because Citi Bike completed its most recent expansion up to 86th Street by November 2015 (CitibikeNYC). This period excludes the lower demands seen on weekends and during the holiday season starting from December 21. During December 1 to 20 there were 466 active stations in Manhattan and Brooklyn, with a total capacity of $\sum_i r_i = 15777$ docks. The number of bikes used is approximately $b = 6074$. The daily number of trips in this period was 31,400, which is lower than the August average of 45,000, presumably because of cold weather. We subsequently adjusted the rates by a multiplier of 1.5 to align our data to an average non-winter month after expansion.

The input data used in the simulation consists of the flow rates between pairs of stations and the trip durations. The flow rates are taken to be piecewise constant over each of the 48 30-minute time intervals throughout the day, and are estimated from the data. The flow rate $\mu_{t,i,j}$ in time interval t, $t = 1,\ldots,48$ from station i to j is calculated from the total number of observed trips from station i to j in that interval, divided by the time that the station is not empty. (This accounts for the censoring that happens when no bikes are available, but does not account for the censoring that happens when a biker tries to return a bike to a full rack and must go to an adjacent station.) The trip durations are obtained using linear regression, fitting the log of the trip durations seen in data to the log of the predicted cycling durations from Google Maps. Figure 1 shows a scatter plot in log scale with the fitted regression line (right plot) on 85% of the data inside the central ellipse (left plot).

2.3 Discrete-Event Simulation Model

We adapt an existing simulation model (O'Mahony 2015) written in Python that operates in discrete time (minute by minute). The arrival process of potential bikers at stations are independent across stations, and at each station i is a time-varying Poisson process with rate $\mu_{t,i} = \sum_j \mu_{t,i,j}$ in time interval t, with the arrival times rounded to the nearest minute. The destination of a biker leaving station i in time interval t has a multinomial distribution with the probability of going to station j estimated by $P_{t,i,j} = \mu_{t,i,j}/\mu_{t,i}$. The associated trip duration T_{ij} is lognormally distributed with parameters obtained from the regression model above, and also rounded to the nearest minute.

The system evolves as follows. In each new minute we generate and schedule the trips starting in that minute from each station i and assign a destination and duration to each trip. Next, all the trips scheduled to arrive or depart a station in this minute are executed as follows. If a departing trip cannot start now because the origin station i is empty, the customer leaves the system and the trip is recorded as a "failed-start". If an arriving trip cannot end because the destination station j is full, the state "failed-end" is triggered, and a new trip heading from j to the nearest station is scheduled. Customers make at most 3 attempts to

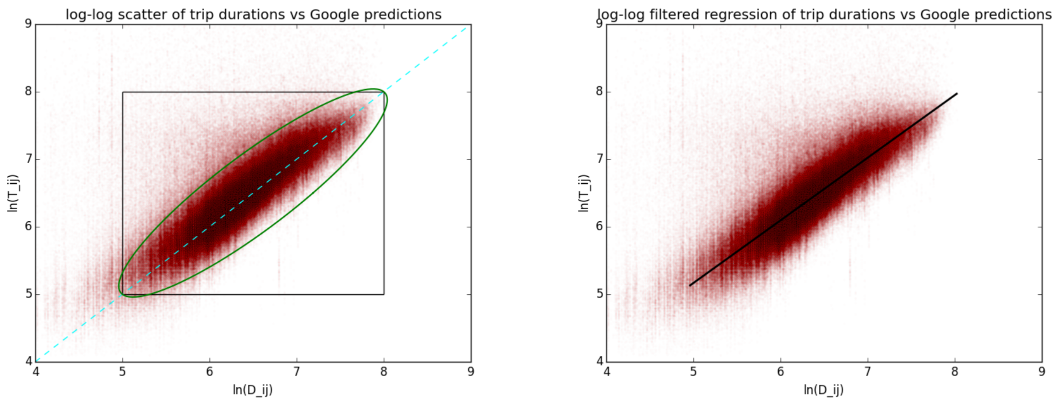


Figure 1: The regression line is $\ln(\text{observed}) = 0.93\ln(\text{google}) + 0.53 + \varepsilon)$, where ε is normally distributed with mean 0 and variance 0.066. The R^2 value of the fit is 0.806. Durations are measured in seconds.

return a bike, at which point we label the final trip as a "bad-end," which happens rarely (less than 1% of the trips). With this simulation model, the objective in (1) is

$$\min_x f(x) = E_x[\#\text{failed-starts}] + E_x[\#\text{failed-ends}] + E_x[\#\text{bad-ends}], \tag{2}$$

where each expectation is estimated using sample-average approximation over a fixed set of replications of the simulation model.

We have sped up the basic model by a factor of approximately 40% by generating trips in 30-minute batches and using conditional uniform occurrences to generate the Poisson arrival processes. The time required to simulate one replication of an 18-hour day (assuming nothing happens from 12-6AM in which only 2.4% of the daily trips occur) is around 1.4 seconds on a desktop with 4-core Intel Core i7-3770 CPU 3.40 GHz processor, 16G memory, and Ubuntu 14.04 OS.

2.4 Starting Solutions for the Simulation Optimization

We introduce three alternative methods to generate starting solutions.

2.4.1 An Equal Allocation Solution

With the current capacity at each station fixed at its true value as of December, 2015, a naïve solution is to allocate some bikes to every station in proportion to its capacity. This solution does not take the flows in and out of stations into consideration, and so suffers from flow imbalances.

2.4.2 A Fluid Model Solution

Now consider flow rates but ignore randomness, so that in period t customers drop off bikes at station i with constant rate $\lambda_{t,i} = \sum_j \mu_{t,j,i}$ and pick up bikes at the same station with constant rate $\mu_{t,i}$. (For simplicity we assume that the trip durations are 0.)

The key idea is to calculate the minimum level of bikes and docks to start the day with so that the objective is 0. To achieve that, suppose the level of bikes in station i at minute q in the day $Y_i(q)$ is not constrained by 0 or the capacity. Then given any starting bike allocation x_i, $Y_i(q)$ is equal to the initial level x_i, plus the net flow of bikes in all the complete 30-minute intervals before q, and the net flow in the last less-than-30-minute interval that contains q. That is,

$$Y_i(q) = x_i + \left(\sum_{t=1}^{\lfloor q/30 \rfloor} 30(\lambda_{t,i} - \mu_{t,i})\right) + (q - 30\lfloor q/30 \rfloor)(\lambda_{q,i} - \mu_{q,i}).$$

To avoid cost when the station is empty or full, in a perfect world we would start with $\hat{x}_i = x_i - \min_q Y_i(q)$ bikes and capacity $\hat{r}_i = \max_q Y_i(q) - \min_q Y_i(q)$ docks so there are sufficient bikes and docks throughout the day. Figure 2 gives an example of this "ideal" solution.

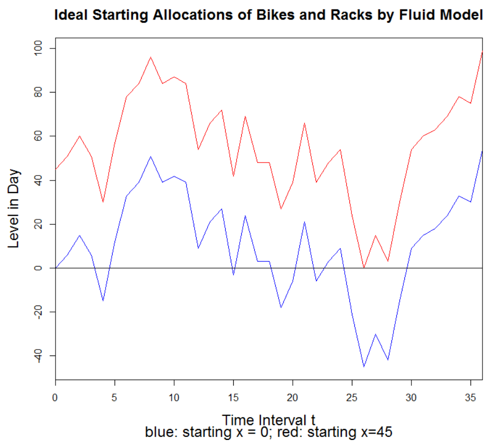


Figure 2: The bike level in an 18-hour day for the fluid model. The blue curve starts with 0 bikes, with lowest and highest levels of -45 and 54. The red curve starts with the "ideal" of 45 bikes and 99 docks.

Starting with the ideal $\hat{x}_i$ and $\hat{r}_i$ for all stations i would require many more bikes and docks than we have. Therefore, after obtaining these ideal levels, we scale the dock allocations to ensure that we don't exceed our available capacity. In doing so, some care is required to account for the integer nature of the dock allocations; we omit the details. A similar scaling is used for the bike allocations. An additional complexity results when these unequal scalings result in a station receiving more bikes than its allocated docks, but again we omit the details of our ad-hoc solution.

A shortcoming of the fluid model, noted in Jian and Henderson (2015), is that it allocates almost no capacity to stations with nearly balanced inward and outward flow rates, irrespective of their magnitude.

2.4.3 A Continuous-Time Markov Chain Solution

Henderson, O'Mahony, and Shmoys (2015) use a very similar model to ours, and show that under the admittedly very strong assumption that the objective function in our optimization problem is separable by stations, i.e., decomposes into a sum of functions, where the function for each station depends only on the inflow and outflow rates and the number of bikes and docks allocated to that station. The change in the allocations at one station does not affect the inflow and outflow rates at its downstream and upstream stations. This dramatic simplification, together with a result establishing that the objective function can be extended to a piecewise-linear convex function, allows them to obtain bike and dock allocations by solving a linear integer program. Their result only applies to the case where the flow rates are constant in time. An extension of their result due to Freund, Henderson, and Shmoys allows us to solve the time-inhomogeneous problem, albeit still under the "objective separable by stations" assumption. The solution thus obtained is our third starting solution for simulation optimization.

3 SIMULATION OPTIMIZATION

In this section we suggest four simulation-optimization heuristics that can be used to solve (1) over different time horizons and problem features. Each alternative is tested using the starting solutions in Section 2.4. The structures of the heuristics are similar. In each iteration we generate a trial solution and evaluate it with the discrete event simulation model in Section 2.3. If the trial solution improves the objective, then we move to the solution; otherwise we stay at the last solution. We use common random numbers to evaluate each trial solution using 30 replications of the simulation. In generating the trial solutions, we treat the simulation as a "white box," exploiting gradient-like information that can be gathered inside the model.

Recall that in the objective function (2), "# failed-starts" is the number of trips that cannot start because the origin station has no bikes, "# failed-ends" is the number of trips that cannot end because the destination station has no empty docks, and "# bad-ends" is the number of trips in which the customer finally abandons the bike after 3 failed attempts. Although "bad-ends" have serious consequences, they happen very rarely ($< 1\%$ of the trips) and thus are ignored in our methods when generating trial solutions. We denote the allocation of bikes (docks) at station i at the beginning of the day by $x(i)$ $(r(i))$, where i is the station id of one of the 466 stations. As a reminder, the total number of bikes in the system is $b = 6074$, and the capacity at each station is the status-quo as of December 2015.

3.1 Simulation Optimization Heuristics to Optimize Bike Allocations

First, consider a simpler problem that only optimizes the bike allocation x for a *fixed* capacity r equal to the status-quo of December 2015 by treating r in (1) as an input parameter. We start with a simple heuristic that optimizes the bike allocation only considering the morning rush (6-10am), similar to the one introduced in Jian and Henderson (2015), then turn to optimizing over an 18-hour day (6-12am). We disregard 12-6am because only 2.4% of the daily trips occur in that interval.

3.1.1 Optimizing Bike Allocations for the Morning Rush-hour

Suppose for now we are only optimizing the objective (2) over the morning rush (6-10am).

From m replications (days) of the 6-10am simulation with x as the initial allocation, we can estimate the objective evaluated at x from the counts of the failed-starts and failed-ends over all replications. Suppose we also obtain the list of *stations* where the failed-starts and first attempts of failed-ends (not including later attempts when the destination station is full) arose. Define the ordered list $statE(x,m,\ell)$ (we will suppress the arguments) as origin stations with the top ℓ # failed-starts, and $statF(x,m,\ell)$ as the destinations with the top ℓ # failed-ends. We expect that the statE stations have many failed-starts because they are too empty at 6am, whereas the statF stations have many failed-ends because they are too full at 6am. Thus, increasing the initial bike level x at a station in statE and decreasing it at a station in statF should decrease the # failed-starts and the # failed-ends in (2). This is mostly true for the morning rush-hour period, but is flawed if we are optimizing over the entire day, as discussed in Section 3.1.2. The method swaps w bikes between two randomly selected stations in these lists as described in Heuristic 1.

We use $m = 30$ replications. The list size for statE and statF is $\ell = 20$. The number of bikes allowed to move from/to each station in each iteration w is changed adaptively along the iterations. Initially $w = 3$, and whenever a consecutive of 100 iterations (100 trial solutions) cannot improve the objective, w is reduced by 1. When starting with a solution closer to optimality (like CTMC), w can be adjusted to start from 1, but here we keep it consistent for all starting solutions just for the fairness of comparison. The initial solution x_0 and the terminating one are evaluated with 50 and 100 independent replications, respectively, to obtain an independent estimate of the objective-function value, independent of the replications used in the search. We choose to stop the heuristic when it fails to improve the objective for 200 consecutive iterations (trial solutions generated).

The heuristic is tested from the proportional-allocation solution and the CTMC solution, giving Figure 3, in which the x-axis is the number of simulated days (replications), and the y-axis is the objective. The equal allocation solution starts with a 95% confidence interval for the objective of 4673 ± 35 and ends at 2775 ± 22. The CTMC solution starts with an objective of 2276 ± 30 and ends at 2236 ± 22. The heuristic makes great improvements to the equal allocation solution (42%), but not much to the better CTMC solution (2%).

Heuristic 1 Optimizing x for the morning rush.

```
Require: A starting solution $x_0$. The list size $\ell$ for statE and statF. The random seed for the simulation
    (daySeed). Number of replications $m$ for the simulation. Number of bikes $w$ allowed to move to/from
    each station in each iteration.
 1: initialize Set $k = 1$. Run simulation to evaluate $x_0$ and obtain the initial statE and statF.
 2: repeat
 3:     procedure GENERATE TRIAL SOLUTION
 4:         Randomly choose $s_E$ from statE and $s_F$ from statF such that $x_{k-1}(s_E) + w \le r(s_E)$ and $x_{k-1}(s_F) -$
    $w \ge 0$.
 5:         Generate trial solution $x'$ by add $w$ bikes to $s_E$ and removing $w$ bikes from $s_F$ based on $x_{k-1}$.
 6:     procedure SIMULATE AND EVALUATE
 7:         Set random seed = seed.
 8:         Evaluate $x'$ using the sample average of the objective in (2) over $m$ replications of simulation.
 9:         if $x'$ shows improvement in the average objective then
10:             Set $x_k = x'$ and let $k = k + 1$. Record statE and statF from the simulation.
11:         else
12:             Go back to generate another trial solution.
13: until Stopped.
```

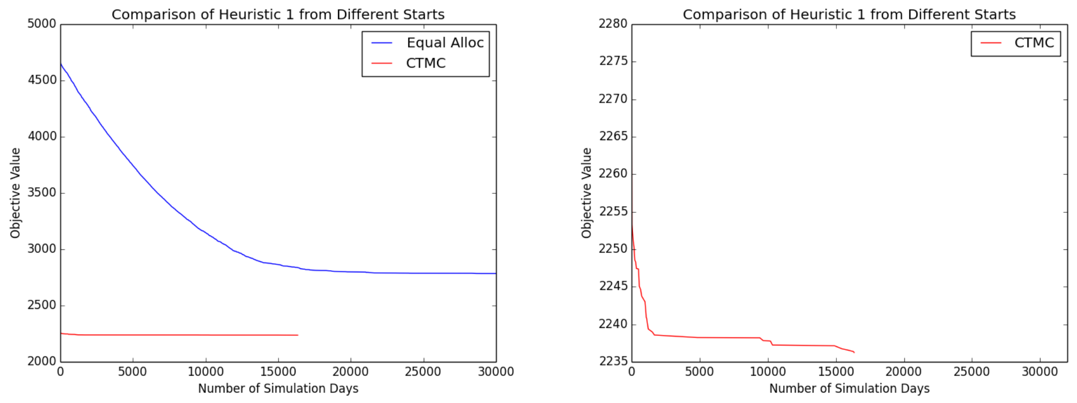


Figure 3: The objective of running Heuristic 1 starting from the equal allocation and CTMC solutions. The left one is the comparison of the two, and the right one is CTMC only on a magnified scale.

3.1.2 Optimizing Bike Allocations for the Entire Day

Consider the same bike-allocation-only problem as above, but instead evaluating the objective over an 18-hour day. When we use Heuristic 1, changing only the simulation period to 18 hours, the method was not able to find improvement for a long time. One of the problems is that most stations behave very differently in the morning and in the afternoon, but statE and statF are still chosen based on the counts of failed starts and failed ends over the *entire* day. If a station in statE has many failed starts in the morning, then adding bikes to the station at the beginning of the day helps in reducing the objective. However, consider a busy station in statE that fills in the morning and then empties in the afternoon. Adding bikes to this station at the beginning of the day makes # failed ends in the morning worse. Meanwhile, because the station starts to empty at the same time irrespective of adding bikes or not, it does not help to reduce the count of failed starts in the afternoon. Indeed, the sample path for the increased allocation of bikes couples with that of the previous allocation once the bike level hits 0 or the capacity. A symmetric problem arises with stations in statF that empty out in the morning and fill up in the afternoon; reducing their bike levels at the beginning of the day would increase the objective instead. Moreover, these stations with both failed starts and failed ends are typically those that have large traffic, e.g., near Penn Station, and thus contribute heavily to the objective.

To adjust Heuristic 1 to address this issue, we define the following 7 types of stations, with illustrations in Figure 4.

1. statEA: the stations that are empty in the morning and not full in the afternoon.
2. statEP: the stations that are empty in the afternoon and not full in the morning.
3. statFA: the stations that are full in the morning and not empty in the afternoon.
4. statFP: the stations that are full in the afternoon and not empty in the morning.
5. statBI: the stations that are full in the morning and empty in the afternoon.
6. statBD: the stations that are empty in the morning and full in the afternoon.
7. statC: the stations that contribute the least to the objective by rarely being full or empty.

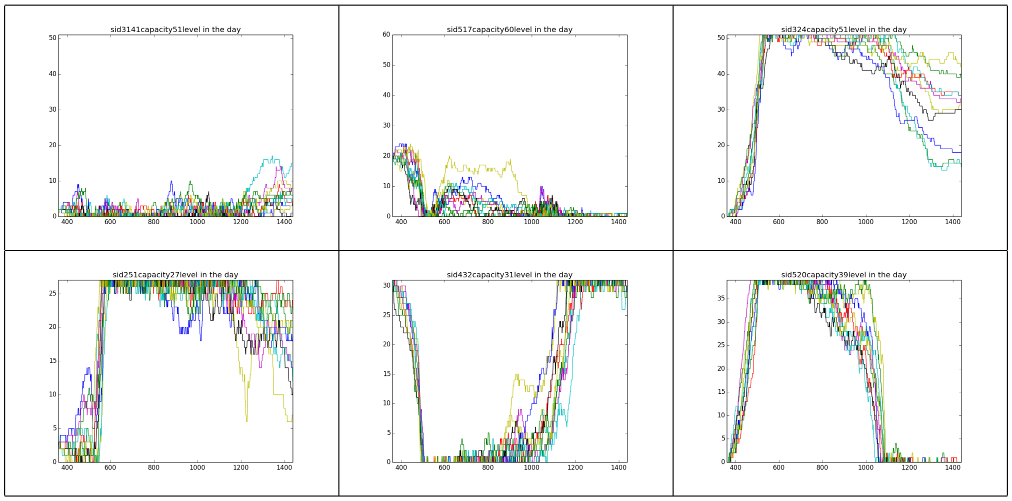


Figure 4: The bike levels for the example stations from the lists statEA, statEP, statFA, statFP, statBI, and statBD (ordered from left to right in rows) over 10 replications. The x-axis is in time from 6-12am.

It appears that statEA, statEP, and statBI need more bikes in the morning, whereas statFA, statFP, and statBD need fewer bikes in the morning. The stations in statC contribute the least to the objective, and thus are used as "back-up." This inspires Heuristic 2 that changes the method for generating trial solutions, while keeping the rest of Heuristic 1 the same.

Heuristic 2 Optimizing x for the entire day.

```
1: procedure GENERATE TRIAL SOLUTION
2:     Randomly choose a station $s_{\text{type}}$ from each of the list types {EA, EP, FA, FP, BI, BD}.
3:     To generate trial solution $x'$ from $x_{k-1}$
4:     take $w$ bikes from each of $s_{FA}$, $s_{FP}$ and $s_{BD}$,
5:     give $w$ bikes to each of $s_{EA}$, $s_{EP}$ and $s_{BI}$.
6:     If any of the movements is not possible because of capacity restrictions, we substitute the station
   by a random station in statC that allows the movement.
```

With the same configurations of m, ℓ, w, and the stopping criteria as in Section 3.1.1, Figure 5 depicts the progress of this heuristic starting from the equal allocation and CTMC solutions. The equal allocation starts with a 95% confidence interval of the objective of 12249 ± 89 and ends at 10428 ± 47 (-15%). The CTMC solution starts at 9239 ± 73 and ends at 9168 ± 46 (-1%). The percentage difference between the starting and ending objectives has decreased compared to when optimizing only over the morning rush, suggesting that the 18-hour day problem with its time-flow complexities is more difficult than the rush-hour problem.

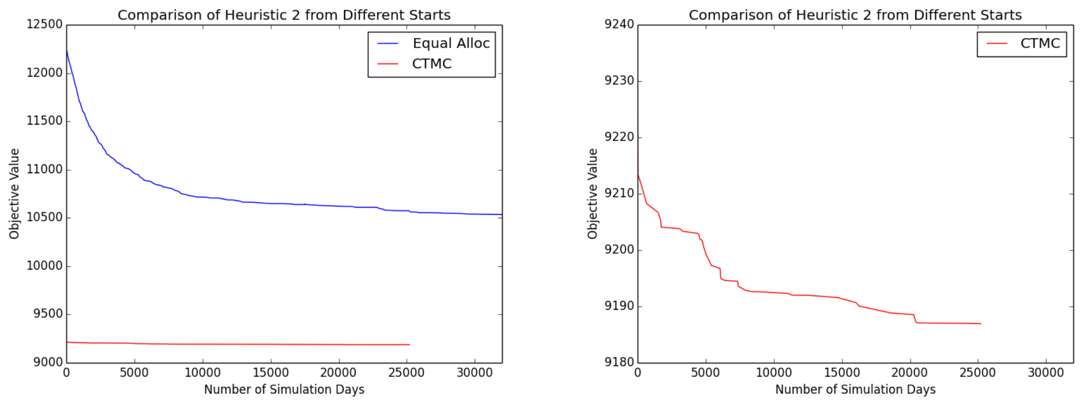


Figure 5: The objective of running Heuristic 2 starting from the equal allocation and CTMC solutions. The left plot is the comparison, and the right one is CTMC only on a magnified scale.

3.2 Simulation Optimization Heuristics to Optimize Both Bike and Dock Allocations

Now we return to the original formulation (1), solving for the allocations of both bikes and docks, initially just over the morning rush and then for an 18-hour day.

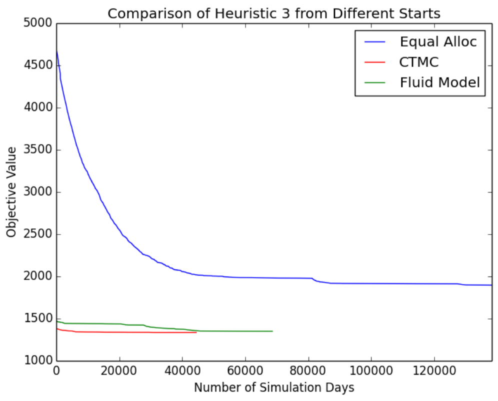
3.2.1 Optimizing Bike and Dock Allocations for the Morning Rush-hour

To incorporate the movements of docks into Heuristic 1, notice that some stations in statE cannot receive more bikes because they are already full. Increasing the capacity at such stations allows us to allocate more bikes and thus reduces the # failed-starts. Similarly, some stations in statF start empty, so increasing the capacity at such stations allows them to receive more bikes and thus reduces the # failed-ends. The docks added to these two types of stations come from statC that is subject to the least amount of change in the objective due to the loss of a dock or a bike. Thus we change the procedure for generating the trial solution in Heuristic 1, giving Heuristic 3.

Heuristic 3 Optimizing x and r for the morning rush.

```
1: procedure GENERATE TRIAL SOLUTION
2:     Randomly choose stations $s_E$ from statE and $s_F$ from statF.
3:     To generate trial solutions $(x', r')$ from $(x_{k-1}, r_{k-1})$,
4:     if $x_{k-1}(s_E) + w \leq r_{s_E}$ and $x_{k-1}(s_F) - w \geq 0$ then
5:         Take $w$ bikes from $s_F$ and give $w$ bikes to $s_E$.
6:     else if $x_{k-1}(s_E) + w > r_{s_E}$ and $r_{k-1}(s_E) + w \leq 60$ then
7:         Take $w$ docks and $w$ bikes from a random station in statC and give them to $s_E$.
8:     else if $x_{k-1}(s_F) - w < 0$ and $r_{k-1}(s_F) + w \leq 60$ then
9:         Take $w$ docks from a random station in statC and give them to $s_F$. Excess bikes will go to $s_E$.
```

This heuristic prioritizes moving bikes first, and if that fails, it moves docks to the stations that cause the failure. With the same configurations of m, ℓ, and w as in Section 3.1.1, Figure 6 gives the progress of this heuristic starting from the equal allocation, the fluid model, and the CTMC solutions. The equal allocation starts with a 95% confidence interval of the objective of 4690 ± 45 and terminates at 1936 ± 18 (-59%), the CTMC solution starts with objective of 1379 ± 28 and ends at 1333 ± 13 (-3%), and the fluid model starts with objective of 1472 ± 32 and ends at 1378 ± 13 (-6%).



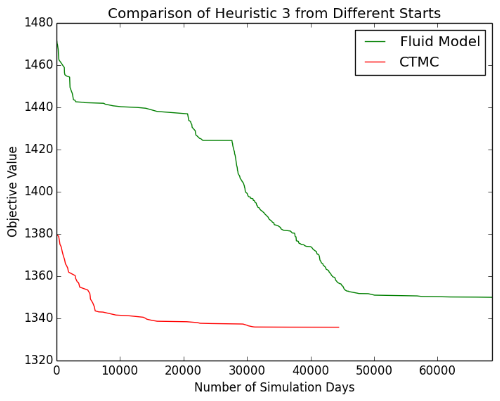


Figure 6: The objective of running Heuristic 3 starting from the equal allocation, the fluid model, and CTMC solutions. The left plot is the comparison of all three, and the right one is the comparison between the CTMC and the fluid model solutions on a magnified scale.

3.2.2 Optimizing Bike and Dock Allocations for the Entire Day

Now we make changes to Heuristic 2, which optimizes the bike allocation over the entire day, to allow for dock movements. Similar to Heuristic 2, the following heuristic requires lists statEA, statEP, statFA, statFP, statBI, statBD, and statC from the last solution (x_{k-1}, r_{k-1}).

With the same configurations of m, ℓ, w, and the stopping criteria as in Section 3.1.1, Figure 7 gives the objective change of this heuristic starting from the equal allocation, the fluid model, and the CTMC solutions. The equal allocation starts with a 95% confidence interval for the objective of 12249 ± 89 and terminates at 8915 ± 47 (-27%), the CTMC solution starts with objective 6937 ± 122 and terminates at 6681 ± 43 (-3%), and the fluid model solution starts with objective 7063 ± 89 and terminates at 6865 ± 44 (-3%).

Heuristic 4 Optimizing x and r for the entire day.

```
1: procedure GENERATE TRIAL SOLUTION
2:     Randomly choose a station $s_{\text{type}}$ from each of the list types in {EA, EP, FA, FP, BI, BD}.
3:     To generate trial solutions $(x', r')$ from $(x_{k-1}, r_{k-1})$,
4:     Move $w$ docks from a random station in statC to $s_{\text{BI}}$ if $r_{k-1}(s_{\text{BI}}) + w \leq 60$.
5:     Move $w$ docks from a random station in statC to $s_{\text{BD}}$ if $r_{k-1}(s_{\text{BD}}) + w \leq 60$.
6:     Try taking $w$ bikes from each of $s_{FA}$, $s_{FP}$ and $s_{BD}$ in $x_{k-1}$. If any of the movements is not allowed
       because the station is already empty, give the station $w$ docks from a random station in statC.
7:     Try giving $w$ bikes to each of $s_{EA}$, $s_{EP}$ and $s_{BI}$ in $x_{k-1}$. If any of the movements is not allowed
       because the station is already full, give the station $w$ docks and $w$ bikes from a random station in statC.
```

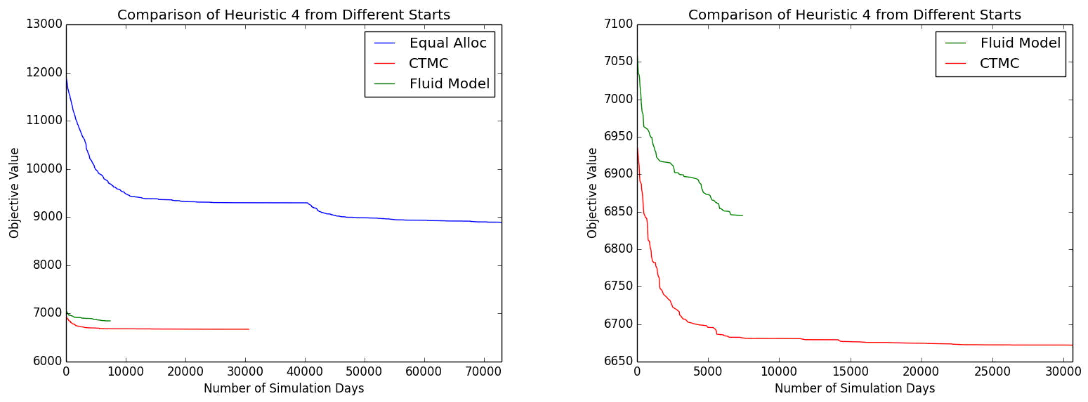


Figure 7: The objective of running Heuristic 4 starting from the equal allocation, the CTMC, and the fluid model solutions. The left plot is the comparison of all three, and the right one is the comparison between the CTMC and the fluid model solutions on a magnified scale.

4 REMARKS

Our heuristics make local improvements to the objective, yielding practically relevant improvements over all solutions, although the improvements relative to the (already apparently well-performing) CTMC solutions are modest. The starting solution plays an important role, because the heuristics cannot close the gap to the best solution we have seen. The heuristics are perhaps best viewed as using approximations for “gradients” (for this integer-variables problem) to guide reallocations of bikes and docks.

All our heuristics move only a few bikes and docks in each iteration, making modest improvements with every 30 replications of the simulation. An alternative is to compute an approximation for the improvements in the objective with regard to all the possible changes ($\pm w$ bikes, $\pm w$ docks, $\pm w$ bikes and docks) to any of the stations in the system, based on simulation results from the current solution. Then we can match up the improvements that reduce the objective the most, and try executing the associated changes while keeping the total number of bikes and docks in the system the same (e.g. $+w$ bikes at a station and $-w$ bikes at another, or $+w$ bikes and docks at a station and $-w$ bikes and $-w$ docks at two other stations). The computation of the impact of potential changes at all stations would be expensive, but this approach could potentially move hundreds of bikes and docks in each iteration. We could follow that large-scale redistribution with one of the heuristics in this paper for finer adjustments.

ACKNOWLEDGMENTS

We thank David Shmoys for his guidance and members of the research team Aaron Ferber, Jaffe Greenwald, Austin Lin and Jasmine Yang for their assistance. The authors would also like to thank Motivate International and New York City Bikeshare for their longstanding cooperation. This work was partially supported by National Science Foundation grants CMMI-1200315, CMMI-1537394, CCF-1526067, and CCF-1522054.

REFERENCES

Chemla, D., F. Meunier, and R. W. Calvo. 2013. “Bike sharing systems: Solving the static rebalancing problem”. *Discrete Optimization* 10 (2): 120 – 146.

CitibikeNYC. Accessed Apr. 28, 2016, http://www.citibikenyc.com/.

Forma, I. A., T. Raviv, and M. Tzur. 2015. “A 3-step math heuristic for the static repositioning problem in bike-sharing systems”. Submitted for publication.

Fricker, C., and N. Gast. 2014. “Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity”. Manuscript.

Fricker, C., N. Gast, and H. Mohamed. 2012. "Mean field analysis for inhomogeneous bike sharing systems". In *Discrete Mathematics and Theoretical Computer Science Proceedings*, 365–376.

Henderson, S. G., E. O'Mahony, and D. B. Shmoys. 2015. "(Citi)Bike Sharing". Submitted for publication.

Jian, N., and S. G. Henderson. 2015. "An introduction to simulation optimization". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, T. M. K. Roeder, C. Macal, and M. Rosetti, 1780–1794. Piscataway NJ: IEEE.

Kim, S., R. Pasupathy, and S. G. Henderson. 2014. "A Guide to SAA". In *Encyclopedia of Operations Research and Management Science*, edited by M. Fu, Hillier and Lieberman OR Series. Elsevier.

O'Mahony, E. 2015. *Smarter Tools for (Citi)bike Sharing*. Ph. D. thesis, Cornell University, Ithaca NY.

Rainer-Harbach, M., P. Papazek, B. Hu, and G. R. Raidl. 2013. "Balancing Bicycle Sharing Systems: A Variable Neighborhood Search Approach.". In *EvoCOP*, edited by M. Middendorf and C. Blum, Volume 7832 of *Lecture Notes in Computer Science*, 121–132: Springer.

Raviv, T., and O. Kolka. 2013. "Optimal inventory management of a bike-sharing station". *IIE Transactions* 45 (10): 1077–1093.

Raviv, T., M. Tzur, and I. Forma. 2013. "Static repositioning in a bike-sharing system: models and solution approaches". *EURO Journal on Transportation and Logistics* 2 (3): 187–229.

Schuijbroek, J., R. Hampshire, and W.-J. van Hoeve. 2013. "Inventory rebalancing and vehicle routing in bike sharing systems". *Tepper School of Business* Paper 1491.

Shu, J., M. C. Chou, Q. Liu, C.-P. Teo, and I.-L. Wang. 2013. "Models for Effective Deployment and Redistribution of Bicycles Within Public Bicycle-Sharing Systems". *Operations Research* 61 (6): 1346–1359.

AUTHOR BIOGRAPHIES

NANJING JIAN is a PhD student in the School of Operations Research and Information Engineering at Cornell University. Her research interests include the convexity detection of black-box functions with noise and applications of simulation optimization to large-scale problems. Her web page is https://people.orie.cornell.edu/nj227. Her email address is nj227@cornell.edu.

DANIEL FREUND is a PhD student in the Center for Applied Mathematics at Cornell University. His research interests include discrete optimization and incentive design, in particular for applications related to sharing and transportation. From June 2015 until May 2016 he worked as a Data Scientist for Motivate International. His web page is https://people.cam.cornell.edu/df365/index.html. His email address is df365@cornell.edu.

HOLLY M. WIBERG is a senior undergraduate student in the School of Operations Research and Information Engineering at Cornell University. Her research interests include simulation optimization and data analytics, with a focus on applications to healthcare. Her email address is hmw54@cornell.edu.

SHANE G. HENDERSON is a professor in the School of Operations Research and Information Engineering at Cornell University. His research interests include discrete-event simulation and simulation optimization, and he has worked for some time with emergency services and bike sharing applications. He co-edited the Proceedings of the 2007 Winter Simulation Conference. His web page is http://people.orie.cornell.edu/∼shane. His email address is sgh9@cornell.edu.